Argonne NATIONAL LABORATORY

Benchmarking Performance of Data Analysis Communication Algorithms

Tom Peterka, ANL, tpeterka@mcs.anl.gov

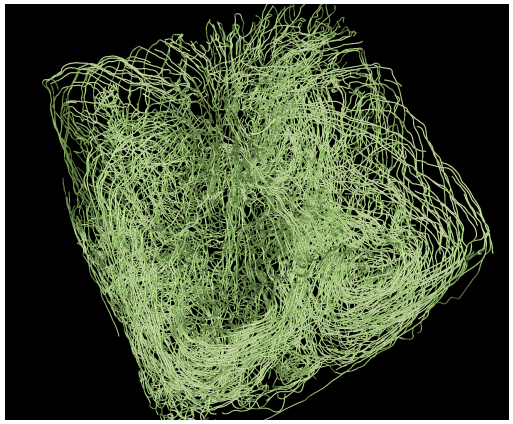
Based on:

Peterka, T., Ross, R.: Versatile Communication Algorithms for Data Analysis. 2012 EuroMPI Special Session on Improving MPI User and Developer Interaction IMUDI'12, Vienna, AT.

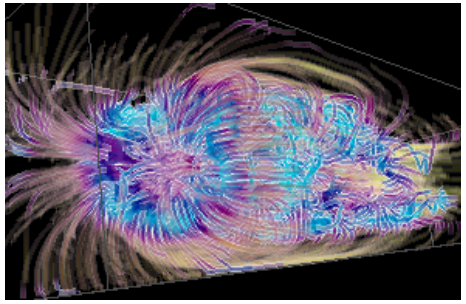
DOE CGF'13
Portland, OR 4/24/13

U.S. DEPARTMENT OF ENERGY

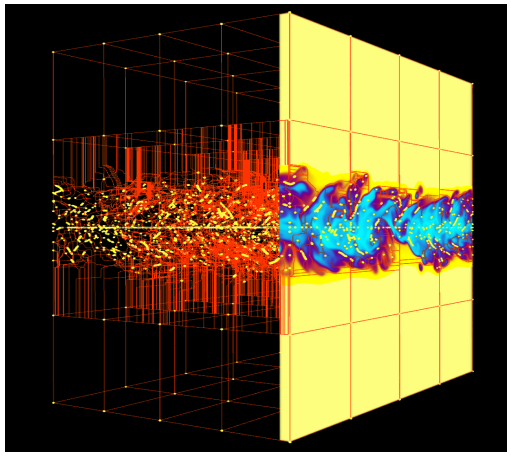
A Few Examples



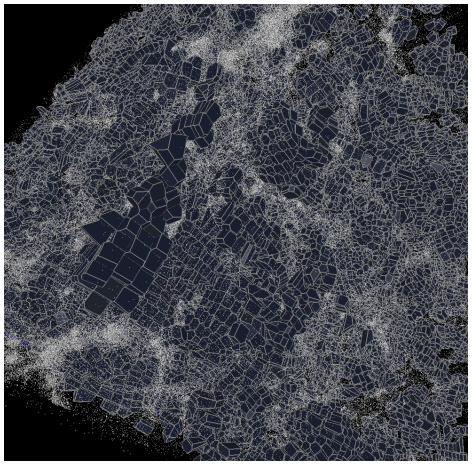
Particle tracing in CFD



Information entropy in astrophysics



Topology in combustion



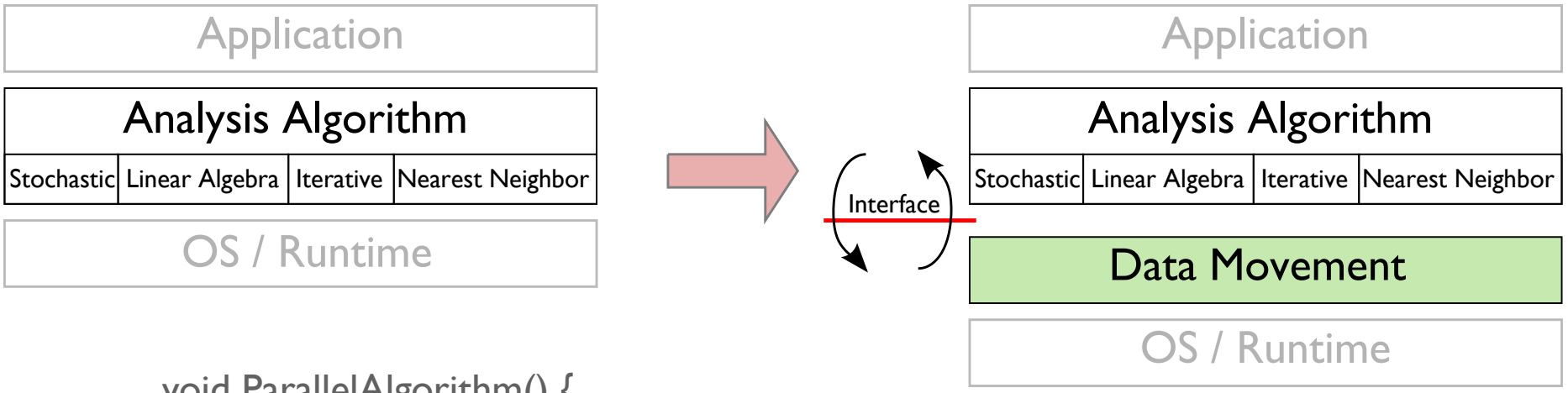
Computational geometry in cosmology

Analysis	Communication
Particle Tracing	Nearest neighbor
Global Information Entropy	Merge-based reduction
Point-wise Information Entropy	Nearest neighbor
Morse-Smale Complex	Merge-based reduction
Computational Geometry	Nearest neighbor
Region growing	Nearest neighbor
Sort-last rendering	Swap-based reduction

Communication pattern for various parallel analysis algorithms

You Have Two Choices to Parallelize Data Analysis

By hand or **With tools**




```
void ParallelAlgorithm() {
   …
   MPI_Send();
   …
   MPI_Recv();
   …
   MPI_Barrier();
   …
   MPI_File_write();
}
```

```
void ParallelAlgorithm() {
   …
   LocalAlgorithm();
   …
   DIY_Merge_blocks();
   …
   DIY_File_write()
}
```

DIY in One Slide

Main Ideas

-Large-scale parallel analysis (visual and numerical) on HPC machines

-Data-parallel problem decomposition

-Scalable data movement algorithms

-Runs on Unix-like platforms, from laptop to supercomputer (including IBM and Cray HPC leadership machines)

Features

-Parallel I/O to/from storage

-Domain decomposition

-Network communication

-Written in C++ with C-style bindings, can be called from Fortran, C, C++

-Autoconf build system

-Lightweight: libdiy.a 800KB

-Maintainable: ~15K lines of code

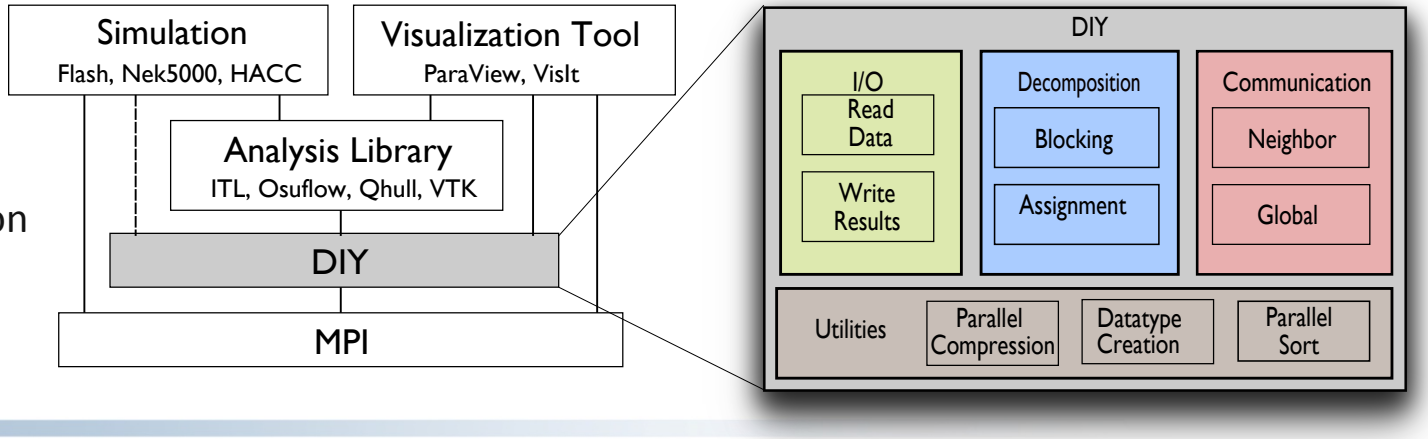
Benefits

-Researchers can focus on their own work, not on parallel infrastructure

-Analysis applications can be custom

-Reuse core components and algorithms for performance and productivity

DIY usage and library organization



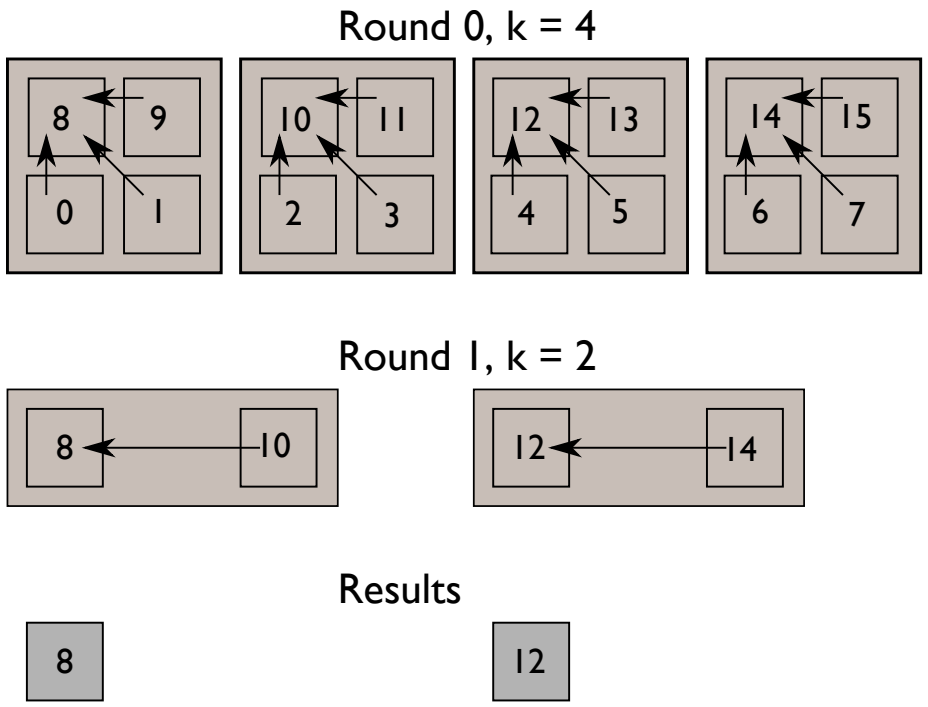
What This Talk is Really About

- Three communication algorithms
  - Global reduction
    - Merge-based
    - Swap-based
  - Neighborhood exchange
- Features beyond MPI
  - Communication among blocks instead of processes (multiple blocks per process)
  - Configurable communication algorithms
    - For performance, eg. variable radix
    - For usability, eg. complete or partial reductions
- Benchmark performance
  - Have demonstrated good performance in applications, but not controlled benchmarks
  - Compare with MPI where applicable (one block per process, complete reductions)

Earlier paper, Peterka et al. LDAV'11, introduced DIY, demonstrated use in two applications. Other papers document performance of analysis applications written using DIY. This talk dives into the three communication algorithms and benchmarks their performance compared to MPI and to earlier applications before DIY.

Merge-Based Global Reduction Algorithm

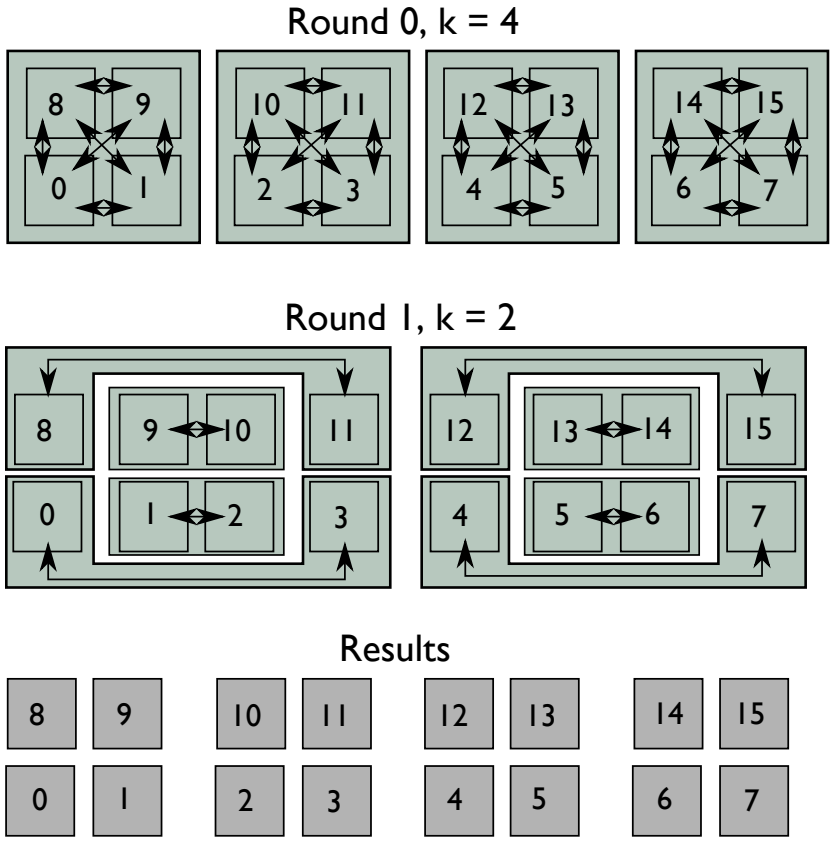
- Associative, not necessarily commutative operations (like MPI)
- Custom datatypes (like MPI)
- For datatypes that cannot be separated automatically (not single buffers) See swap-based reduction for that case
- Communication among blocks not processes (unlike MPI)
- Total and partial reductions (unlike MPI)
- Configurable radix (unlike MPI)



Example of partial merge-based reduction of 16 blocks in 2 rounds.

Swap-Based Global Reduction Algorithm

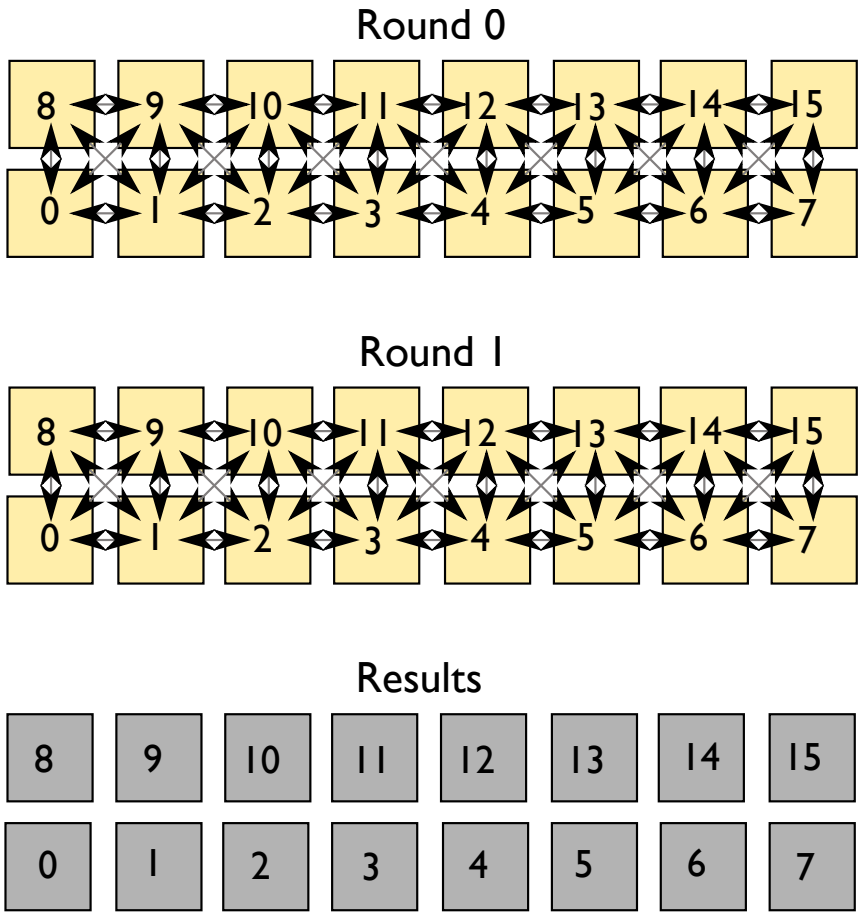
- Associative, not necessarily commutative operations (like MPI)
- Custom datatypes (like MPI)
- For datatypes that can be separated automatically (single buffers)
- Communication among blocks not processes (unlike MPI)
- Total and partial reductions (unlike MPI)
- Configurable radix (unlike MPI)



Example of partial swap-based reduction of 16 blocks in 2 rounds.

Neighborhood Exchange Algorithm

- Not associative, order data-dependent operations (not in MPI)
- Custom datatypes (like MPI)
- Communication among blocks not processes (unlike MPI)
- Adjustable synchronization (wait for some fraction of pending messages in every round)



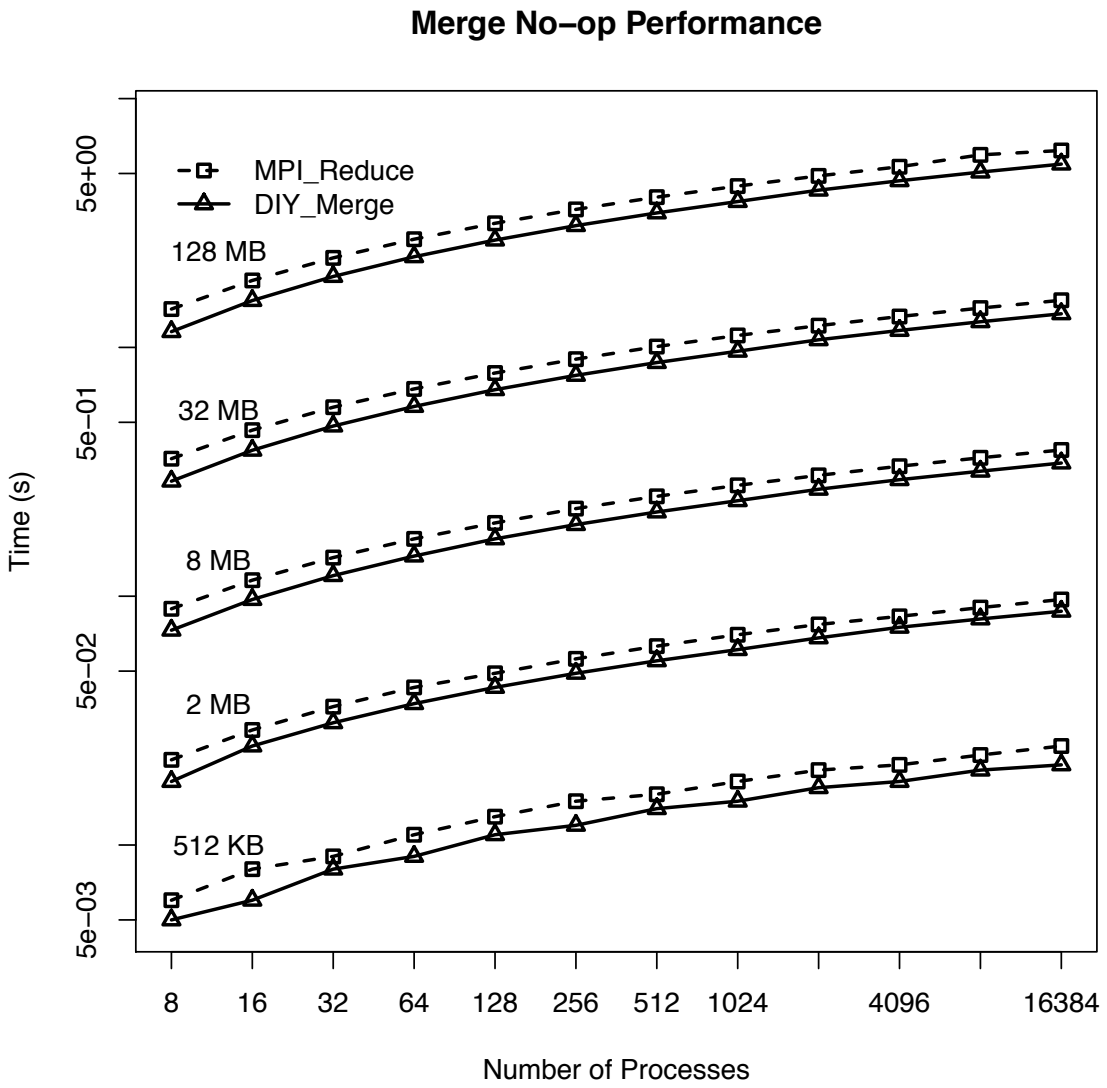
Example of neighborhood exchange of 16 blocks in 2 rounds.

Performance Tests

- Platform
  - Intrepid IBM Blue Gene/P (557 TF)
  - IBM xclxx_r compiler
  - -O3 –qarch=450d –qtune=450
- Reduction test
  - 1 block per process (in order to compare w/ MPI)
  - Complete reduction (ditto)
  - Composite operator, linear combination of two pixel values [r,g,b,a] modulated by a (in order to compare with radix-k, Peterka et al. SC10)
  - Message sizes typical of images being composited (compare w/ radix-k)
  - *smp* (1 process per node) mode
- Neighborhood exchange test
  - Parallel particle tracing of vector field (compare w/ Peterka et al. IPDPS'11)
  - 2048^3 grid points
  - 256K particles
  - 1000 advection steps
  - *vn* (4 processes per node) mode

Merge-Based Communication-Only Performance

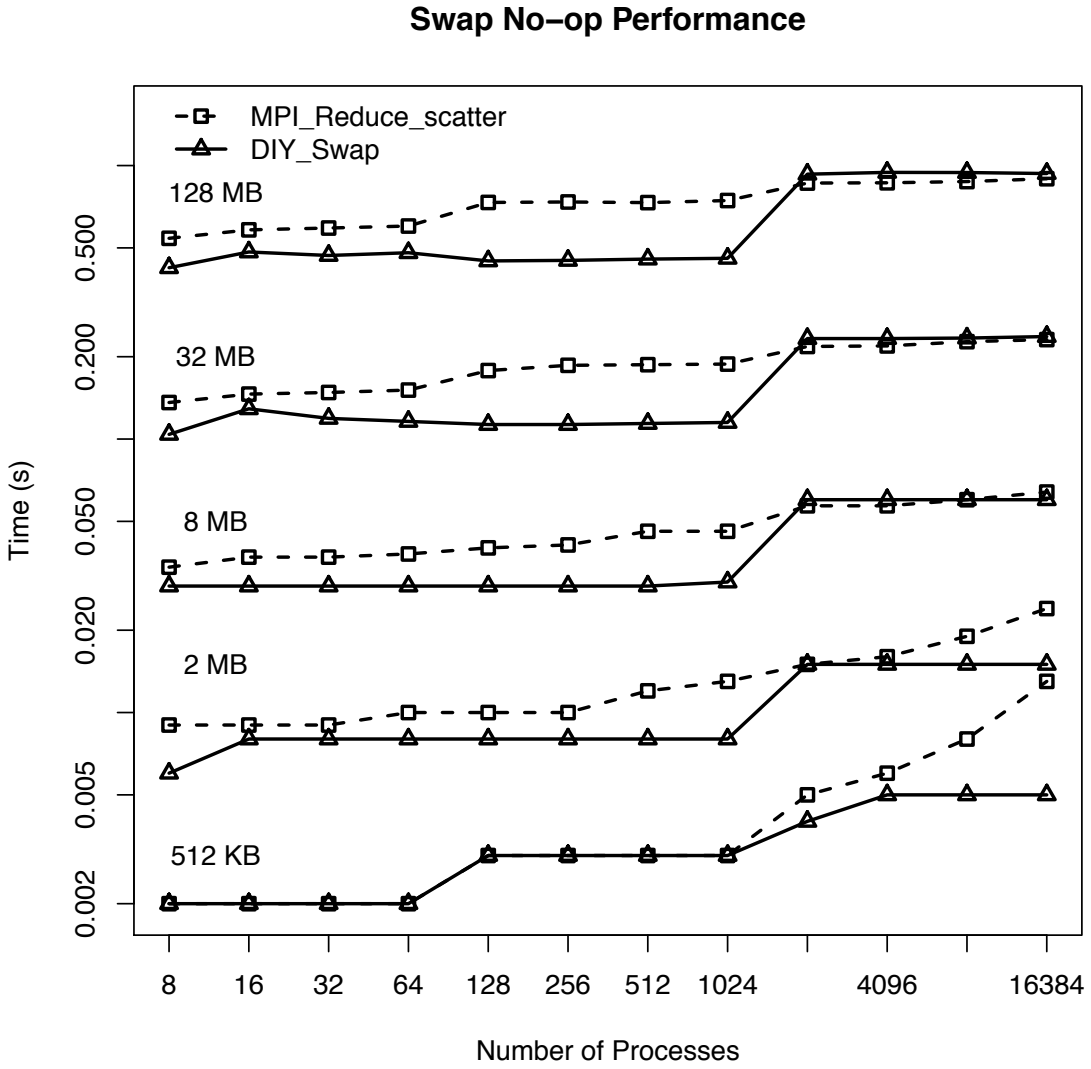
- No-op
- Radix = 2 worked best
- 10% faster than MPI implementation of reduce



Benchmark of DIY merge-based reduction vs. MPI reduce

Swap-Based Communication-Only Performance

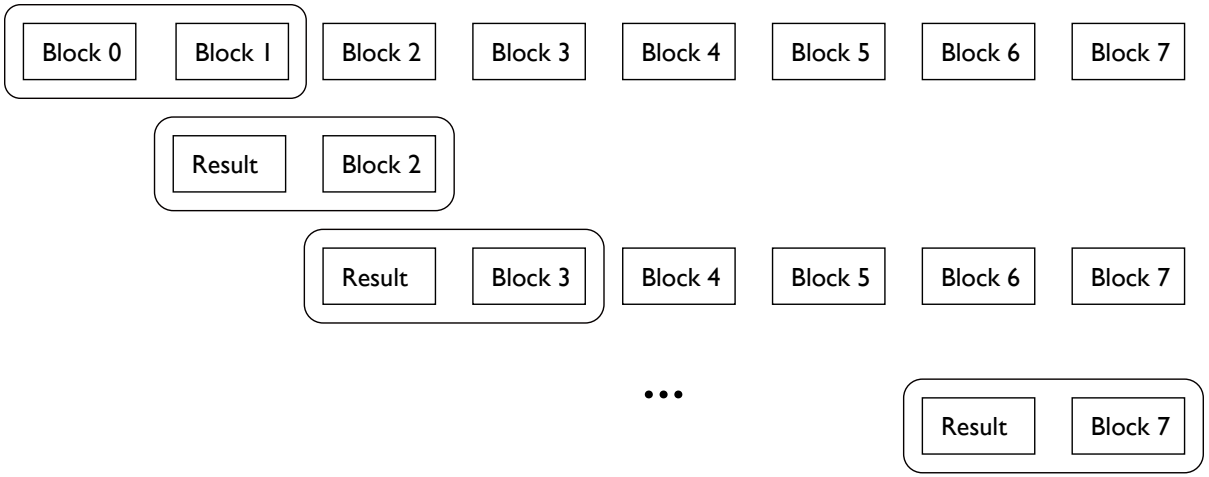
- No-op
- Radix = 8 worked best
- Up to 60% faster than MPI implementation of reduce-scatter (at 1024 processes)



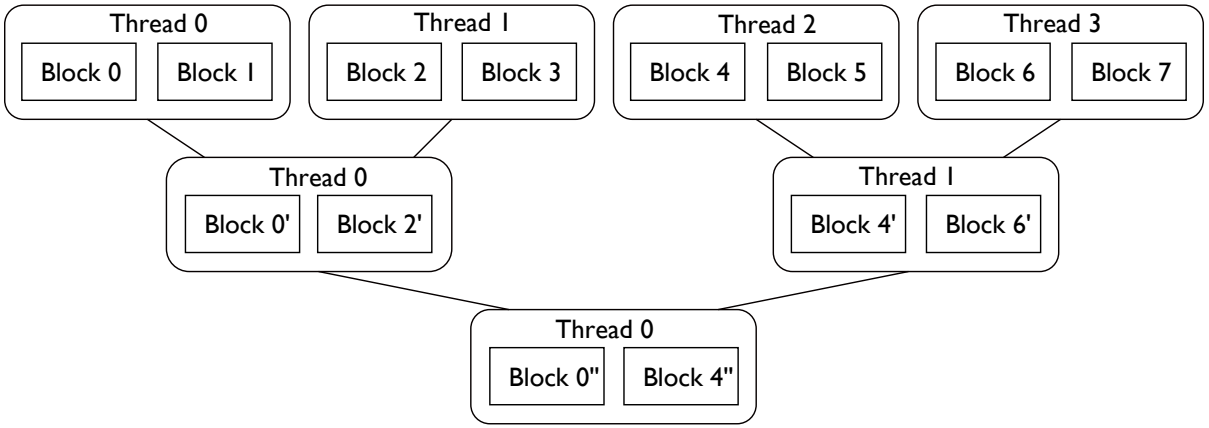
Benchmark of DIY swap-based reduction vs. MPI reduce-scatter

Higher Radix Exposes Additional Parallelism Opportunities

- Having more blocks to reduce allows local multithreading
- Maintain order for noncommutative operations
- Tree reduction as in Moreland et al. SC11



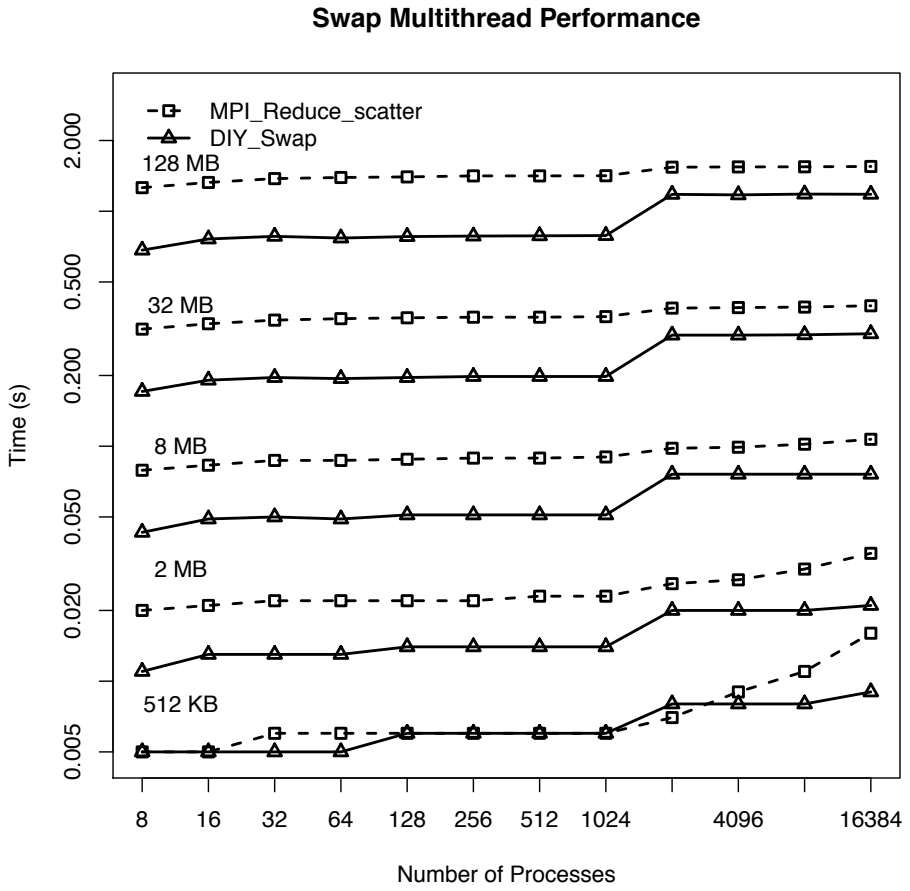
Single thread local serial reduction of eight blocks



Multithread local tree reduction of eight blocks and four threads

Swap-Based Communication + Multithread Reduction Performance

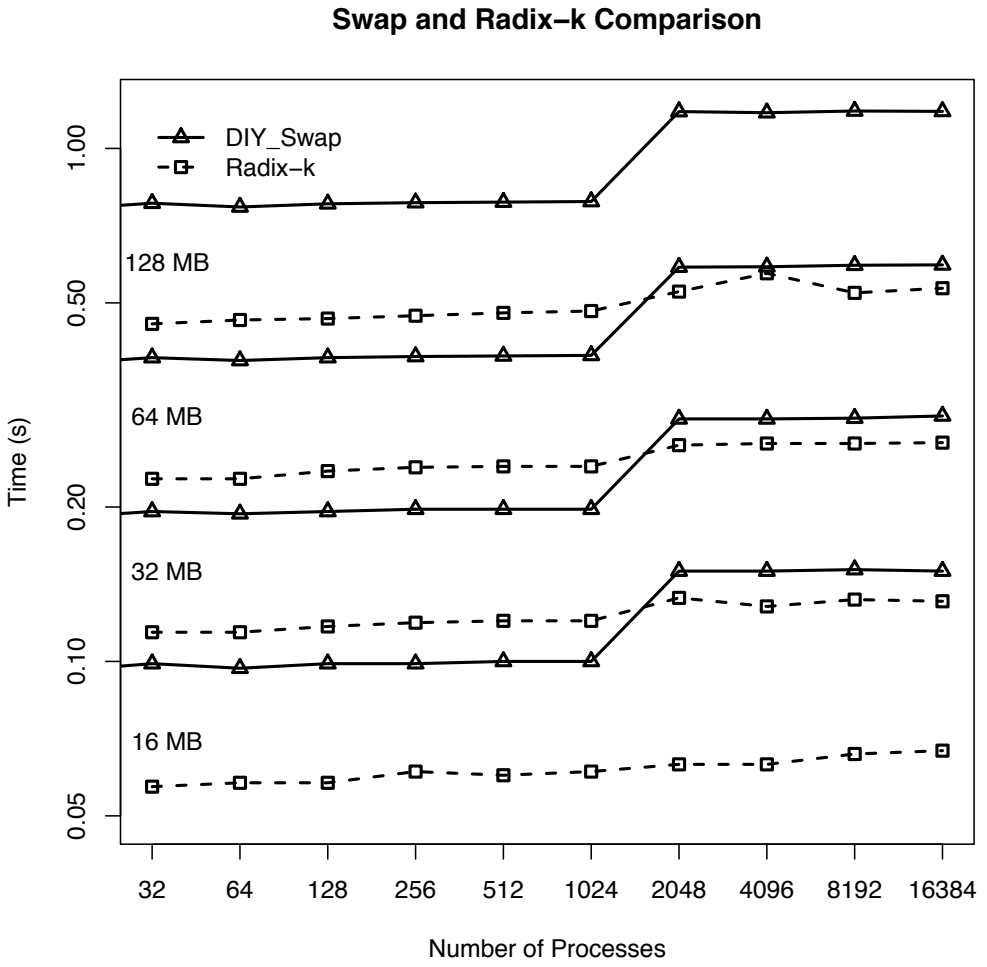
- Image compositing op
- radix = 8
- Openmp threads = 4
- Up to 1.8X faster at 1024 processes



Benchmark of DIY swap-based multithread reduction vs. MPI reduce-scatter

DIY Compared to Radix-k

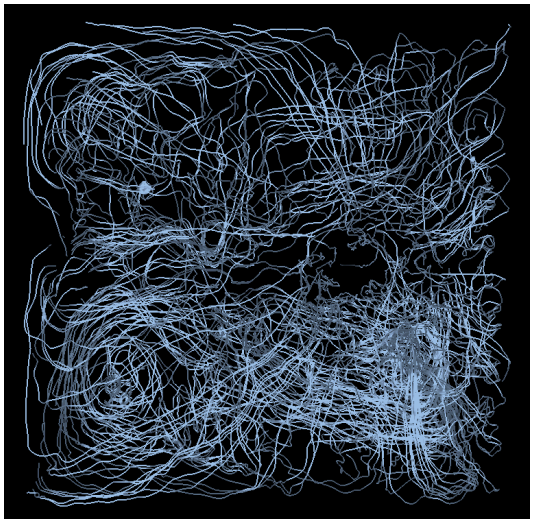
- Image compositing op
- radix = 8
- Openmp threads = 4
- Still 2X slower
- Cause: allowing custom user-defined operations and datatypes requires communication to be done before calling user-defined operator
- We are working to relax this requirement, but results show we still have some work to do.



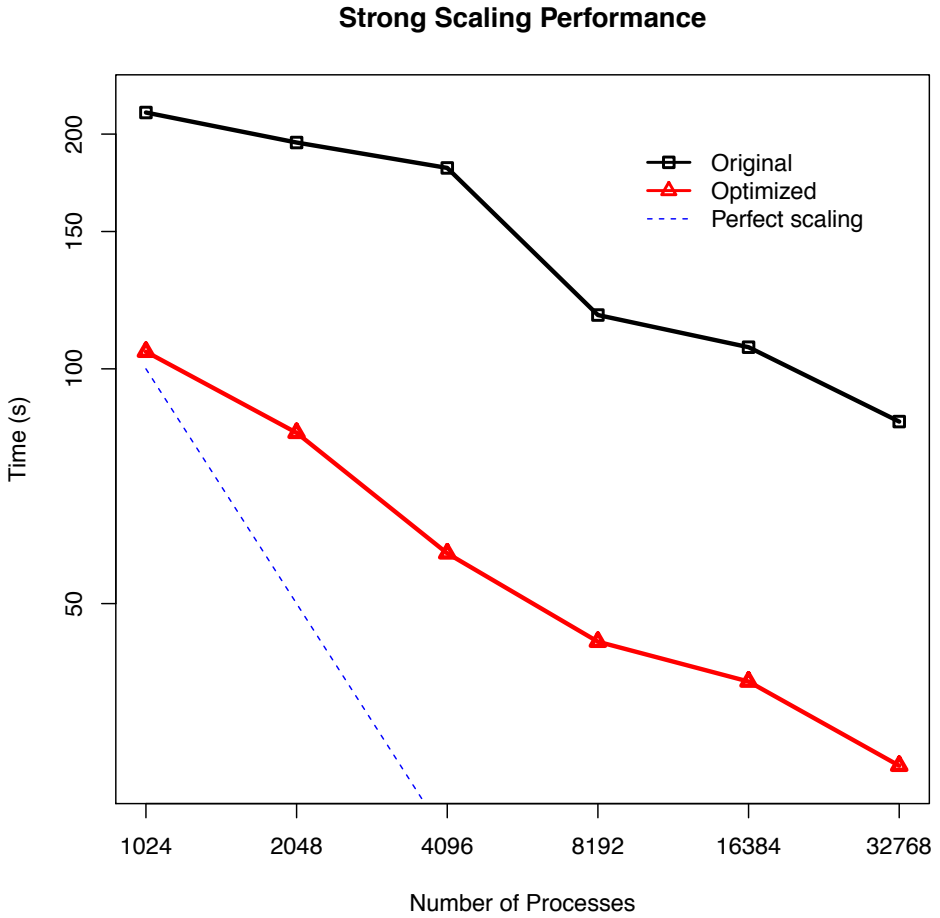
Benchmark of DIY swap-based multithread reduction vs. radix-k

Ending on a Positive Note: Neighborhood Exchange Performance

- Actual application, not a benchmark like reductions
- Comparison of DIY neighborhood exchange with our earlier algorithms
- Primary reason: adjustable synchronization



Particles traced in thermal hydraulics flow field.



Strong scaling for compute + communicate time is a 3X improvement over our earlier algorithm.

Wrapping Up

- Successes
  - Usability
    - Multiple blocks per process; communication between blocks instead of processes
      - Important for fine-grain decomposition, load balancing, portability
    - Partial reductions are a natural outcome of configurable rounds and radices
      - Important for some problems where a full reduction doesn't fit in memory (eg. merge reduction in topological analysis grows in size with each round)
  - Performance
    - Where comparable MPI functions do exist, we perform as well or slightly better
    - Higher radix exposes more parallelism opportunities (threading)
    - Compared to previous algorithms
      - Swap-based reduction compared to radix-k 2X slower
      - Neighborhood communication 3X faster
- Ongoing
  - Implement more overlap for communication and reduction operator
  - Approach radix-k performance for swap reduction

Benchmarking Performance of Data Analysis Communication Algorithms

Tom Peterka, ANL, tpeterka@mcs.anl.gov

DIY can be downloaded from

https://svn.mcs.anl.gov/repos/diy/trunk

We gratefully acknowledge the use of the resources of the Argonne Leadership Computing Facility at Argonne National Laboratory. This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. Work is also supported by the DOE Office of Science, Advanced Scientific Computing Research award No. DE-FC02-06ER25777, program manager Lucy Nowell.

DOE CGF'13

Portland, OR 4/24/13